

# **METHOD, SYSTEM, AND COMPUTER-READABLE MEDIUM FOR MANAGING A HOST SESSION ON A REMOTE COMPUTER**

## **Technical Field**

5           The present invention is related to computer host session management. More particularly, the present invention is related to managing a host session on a remote computer in a computer system.

## **Background of the Invention**

10           Some computer systems utilize a mainframe or host server computer for supporting a number of simultaneous users in a computer network. Users access data on the host server through connected computer terminals over the network. In order to access the data on the host server computer, the terminal computers are used to connect to and communicate with the host server over the network using a terminal emulation  
15           software program such as Telnet. The Telnet program runs on each terminal computer and enables the terminal computer to connect to the host server to establish a host session. Once a host session is established, users may enter commands through the Telnet program which will then be executed as if the user were entering them directly on a screen of the host server. For example, when logging on to the host server to  
20           establish a host session the terminal emulation software may be programmed to expect the password field to start at a specific character position on the host logon screen. Thus, once the appropriate password is received by the terminal emulation software, the password is entered as data in the corresponding password field on the host logon screen of the host server.

25           Periodically, the software running on the host server computer may be updated or modified resulting in changes in formatting of the host screens. These changes may affect the manner in which commands from a connected terminal computer are interpreted such that data specified for a previously defined field on a host screen may no longer be valid for that field. As a result, the host server may no longer recognize  
30           the command from the terminal computer and returns an error to the connected terminal

computer. For example, the host server will return an error if the password has changed such that a current password entered by the terminal emulation program is no longer valid.

One previous method for maintaining screen changes on host servers requires  
5 modifying and recompiling the code in the terminal emulation program. However, modifying and recompiling the terminal emulation program code is often a time-consuming process and requires additional testing, which is often a time-consuming process requiring skilled programming knowledge. Moreover, this method may also require additional testing of the recompiled code to verify that the appropriate  
10 modifications have been made. Another previous method for maintaining screen changes requires the use of "screen scraper" programs in which the actual host screens are recorded to capture the changes. However, in most cases such "screen scraper" programs are platform dependent, that is, they require a specific computer operating system in which to operate. Furthermore, the use of "screen scraper" programs requires  
15 a user to manually review each captured host screen to locate where the error has occurred making their use inconvenient during automated or mechanized processes.

It is with respect to these considerations and others that the present invention has been made.

### **Summary of the Invention**

20 In accordance with the present invention, the above and other problems are solved by methods for managing a host session on a host server or remote computer in which the host screens are predefined in properties files accessible by a terminal or client computer in a computer system. When host screen changes are detected during a host session, an error is generated and data associated with the error is presented to a  
25 user in an errors file. The user may then correct the error by simply modifying or updating the properties files associated with the host screens.

According to one method, a request is sent to establish a host session from a client computer. The request includes a presentation space for displaying screen data. The client computer has access to properties files defining the screens for the host

session. After the request is sent from the client computer, a response to the request is received in the presentation space from the remote computer. The response includes host screen data. Then, the response from the remote computer is identified by comparing the host screen data in the presentation space to screen data defined in one or more of the plurality of properties files for the host session. Finally, an action is performed based on the identified response.

The properties files may include one or more screen properties files for defining the screen data for the host session. Each screen properties file may include a responses section which may include a response type for the response and identifying text for the response. The response type may be success, analyze, or reject. The step of identifying the response by comparing the host screen data in the presentation space to screen data defined in one or more properties files for the host session may include determining the response type for the response by comparing the host screen data to the identifying text defined for the response in the responses section in a screen properties file. The step of performing an action based on the identified response may include processing the response if the response type is success, or printing the presentation space to an errors file if the response type is reject. The properties files may be Java properties files and the host session may be a TN3270 host session.

In accordance with other aspects, the present invention relates to a computer system for managing a host session. The system includes a remote computer and a client computer in communication with the remote computer in the computer system. The client computer includes a memory device for storing a program file and properties files for defining screens comprising screen data for the host session. The client computer also includes a processor, functionally coupled to the memory device, which is responsive to computer-executable instructions contained in the program file stored in the memory device. The processor is operative to send a request to the remote computer to establish the host session, receive in a presentation space a response to the request from the remote computer, identify a response type for the response, and perform an action based on the response type. The response includes host screen data and the response type is defined in one or more of the properties files.

Aspects of the invention may be implemented as a computer process, a computing system, or as an article of manufacture such as a computer program product or computer-readable medium. The computer program product may be a computer storage media readable by a computer system and encoding a computer program of instructions for executing a computer process. The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process.

These and various other features as well as advantages, which characterize the present invention, will be apparent from a reading of the following detailed description and a review of the associated drawings.

### **Brief Description of the Drawings**

FIG. 1 illustrates a computer network architecture which may be utilized in various embodiments of the invention.

FIG. 2 illustrates a computer system architecture of a client computer utilized in various embodiments of the invention.

FIG. 3 illustrates the common properties file illustrated in the computer system architecture of FIG. 2, according to an embodiment of the invention.

FIG. 4 illustrates a screen properties file illustrated in the computer system architecture of FIG. 2, according to an embodiment of the invention.

FIG. 5 illustrates the fields section illustrated in the screen properties file illustrated in FIG. 5, according to an embodiment of the invention.

FIG. 6 illustrates the responses section illustrated in the screen properties file illustrated in FIG. 5, according to an embodiment of the invention.

FIG. 7 illustrates logical operations performed in the computer network of FIG. 1 for managing a host session according to an embodiment of the invention.

FIG. 8 illustrates a screenshot of a presentation space generated by the program file illustrated in FIG. 2, according to an embodiment of the invention.

FIG. 9 illustrates the contents of the common properties file illustrated in FIG. 4 according to one embodiment of the invention.

FIG. 10 illustrates the contents of the screen properties file illustrated in FIG. 5, according to one embodiment of the invention.

### **Detailed Description of the Invention**

5           Embodiments of the present invention provide methods, a system, and a computer-readable medium for managing a host session on a remote computer in a computer system. In the following detailed description, references are made to the accompanying drawings that form a part hereof, and in which are shown by way of illustration specific embodiments or examples. Referring now to the drawings, in which  
10   like numerals represent like elements through the several figures, aspects of the present invention and the exemplary operating environment will be described.

FIG. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Those skilled in the art will appreciate that the invention may be  
15   practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed  
20   computing environment, program modules may be located in both local and remote memory storage devices.

Turning now to FIG. 1, an illustrative computer network architecture for practicing the various embodiments of the invention will now be described. The computer network includes a terminal or client computer 2 operative to execute one or  
25   more application programs. The client computer 2 communicates with a remote or host server computer 2 through the network 18.

Turning now to FIG. 2, an illustrative computer architecture for the client computer 2 (which was discussed briefly above) for practicing the various embodiments of the invention will be described. The client computer 2 may be a standard personal  
30   computer operative to execute one or more terminal application programs, such as

application program 26, for establishing a host session on a remote computer, such as the host server computer 2 described above in FIG. 1.

Alternatively, the client computer 2 may include another type of computing device operative to access a network 18, such as a personal digital assistant or other type of computer. The computer architecture shown in FIG. 2 illustrates a conventional personal computer, including a central processing unit 4 ("CPU"), a system memory 6, including a random access memory 8 ("RAM") and a read-only memory ("ROM") 10, and a system bus 13 that couples the system memory 6 to the CPU 4.

The client computer 2 further includes a mass storage device 14 for storing an operating system 16, the application program 26, a program file 28, a common properties file 30, and screen properties files 32A and 32B, respectively. The application program 26 may be a terminal emulation program for connecting to remote mainframe computers such as the host server computer 2 described above in FIG. 1. As is known by those skilled in the art, terminal emulation programs typically run on local or client computers and enable these computers to connect to and control one or more remote host servers in a computer network. Typically, a connection is made by entering a valid username and password through the terminal emulation program to establish a "host session" with the host server. Once a host session with the host server is established, commands may be entered through the terminal emulation program which will be executed as if they were entered directly on the server console.

In one embodiment the application program 26 is a "tn3270" terminal emulation program for emulating an IBM 3270 computer terminal which is normally used to communicate with IBM host or mainframe computers over a network. Of course, it will be appreciated that the application program 26 may be used to emulate other terminals as well or may alternatively be a "Telnet" program which is a terminal emulation program for TCP/IP networks such as the Internet. Such alternative terminal emulators are known to those skilled in the art.

The mass storage device 14 also includes the program file 28 which receives and sends commands from the application program 26 to the host server 4. In one embodiment, the program file 28 is a Java program file which contains classes and

methods that may be accessed and controlled by the application program 26. The classes and methods include connecting to the host server (i.e., establishing a host session), sending commands to the host server, waiting for a response from the host server, analyzing the host server response, returning the host results to the application  
5 program 26, and/or disconnecting from the host. As will be understood by those skilled in the art, the program file 28 may utilize a Java classes file (not shown) containing Java API and system objects and a number of methods which enable a Java process to connect and communicate with a mainframe host session and to send and receive data streams through the connections. The program file 28 utilizes the common properties  
10 file 30 and the screen properties files 32A and 32B to manage a host session. In one embodiment, the common properties file 28 and the screen properties files 32A and 32B are Java properties files. The program file 28, the common properties file 30, and the screen properties files 32A and 32B will be described in greater detail below with respect to FIGS. 4-10.

15 The mass storage device 14 is connected to the CPU 4 through a mass storage controller (not shown) connected to the bus 13. The mass storage device 14 and its associated computer-readable media, provide non-volatile storage for the client computer 2. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, it should be  
20 appreciated by those skilled in the art that computer-readable media can be any available media that can be accessed by the client computer 2.

By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and non-volatile, removable and non-removable media implemented in any  
25 method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other solid state memory technology, CD-ROM, DVD, or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or

any other medium which can be used to store the desired information and which can be accessed by the computer.

According to various embodiments of the invention, the client computer 2 may operate in a networked environment using logical connections to remote computers, such as the host server computer 4, through the network 18. The client computer 2 may connect to the network 18 through a network interface unit 20 connected to the bus 13. It should be appreciated that the network interface unit 20 may also be utilized to connect to other types of networks and remote computer systems. The client computer 2 may also include an input/output controller 22 for receiving and processing input from a number of devices, including a keyboard, mouse, or electronic stylus (not shown in FIG. 1). Similarly, an input/output controller 22 may provide output to a display screen 23, a printer, or other type of output device. Although not specifically described herein, it should be understood that the host server computer 4 described in FIG. 1 above may also include many of the same components described above with respect to the client computer 2.

FIG. 3 illustrates the common properties file 30 briefly discussed above in the description of the computer system architecture of FIG. 2, according to an embodiment of the invention. As discussed above, the common properties file 30 is utilized by the program file 28 for managing a host session. During a host session, a group of screens are utilized to communicate information between the client computer 2 and the host server computer 4. For example, prior to establishing a host session, a logon screen is used to receive valid user name and password data. This data is then sent to the host server computer which returns a response indicating whether or not the logon was successful. One or more screens are then utilized to display the results of the logon attempt. All of the screens of a host session are predefined in the common properties file 30. For a particular screen, the common properties file 30 defines fields for the screen name 40, the number of data pulses to expect for the screen, the identifying text on the screen, the search method used by the program file 28 for locating the identifying text on the screen, and the starting and ending positions on the screen where the identifying text is located.



FIG. 9 shows the contents of an illustrative common properties file called "Session.Properties" for a host session. As shown in FIG. 10, the properties file defines three screens ("Screen 1", "Screen 2", and "Screen 3") defined for use during a host logon session. For example, "Screen 2" has a screen name entitled "Ready\_For\_Logon" and has a data pulse value of 2. The identifying text for Screen 2 is "Is Now Ready For Logon" and the identifying text has a starting position of 80 and an ending position of 111 on the screen.

FIG. 4 illustrates the screen properties file 32A briefly discussed above in the description of the computer system architecture of FIG. 2, according to an embodiment of the invention. As discussed above, the screen properties files 32A and 32B are utilized by the program file 28 in conjunction with the common properties file 30 to manage a host session. Each individual screen of a host session is defined in a screen properties file. As shown in FIG. 3, the screen properties file 32A includes a fields section 51 for defining screen fields and a responses section 53 for defining responses from the host server 4. The fields section 51 and the responses section 53 will be described in greater detail below with respect to FIGS. 5-7.

FIG. 5 illustrates the fields section 51 briefly described above in the description of the screen properties file 32A illustrated in FIG. 4, according to an embodiment of the invention. The fields section 51 includes definitions for fields appearing on a host session screen. As shown in FIG. 5, the fields section includes the name of each field on the screen (field name 62), the starting and ending positions on the screen where the field is to be found (starting position 64 and ending position 66), and a flag 68 indicating whether the field is protected (read only) or unprotected (read/write).

FIG. 6 illustrates the responses section 53 briefly described above in the description of the screen properties file 32A illustrated in FIG. 4, according to an embodiment of the invention. The responses section 53 includes screen definitions for responses from the host server 4 appearing on a host session screen. As shown in FIG. 6, the responses section includes a response type 72. There are three types of responses which may be returned during a host session: success (indicating that a process may continue), reject (indicating that a process may not continue), and analyze (indicating

that a process determine the appropriate response). Each response defined in the responses section 53 also includes the identifying text for the response 74, the search type 75 used for finding the identifying text on the screen, and the starting 77 and ending 79 positions on the screen where the identifying text is to be found.

5           FIG. 10 shows the contents of an illustrative screen properties file called "Security\_Signon.Properties" for a screen in a host session. As shown in FIG. 10, the screen properties file defines individual fields for the screen name, date stamp, time stamp, and a command line which are defined in the fields section 51. Also as shown in FIG. 10, the screen properties file defines response types "Success1, Reject1, and  
10   Reject2" as well as the identifying text associated with each response.

          FIG. 7 illustrates logical operations 700 performed in the computer network of FIG. 1 for managing a host session according to an embodiment of the invention. The logical operations of the various embodiments of the present invention are implemented  
15   (1) as a sequence of computer implemented acts or program modules running on a computing system and/or (2) as interconnected machine logic circuits or circuit modules within the computing system. The implementation is a matter of choice dependent on the performance requirements of the computing system implementing the invention. Accordingly, the logical operations making up the embodiments of the present invention described herein are referred to variously as operations, structural devices,  
20   acts or modules. It will be recognized by one skilled in the art that these operations, structural devices, acts and modules may be implemented in software, in firmware, in special purpose digital logic, and any combination thereof without deviating from the spirit and scope of the present invention as recited within the claims attached hereto.

          The logical operations 700 of FIG. 7 begin at operation 702 where the application program 26 initiates a request to connect the client computer 2 to the host  
25   server 4 to establish a host session. The request is sent to the program file 28 which sends the request to the host server 4 at operation 704. As discussed above in the description of FIG. 1, in one embodiment, a Java classes file containing Java API and system objects and a number of methods is utilized to enable the program file 28 to  
30   connect and communicate with the host server 4 and to send and receive data streams

through a network connection. In this embodiment, the Java classes file sends and receives data between the program file 28 on the client computer 2 and the host server 4.

5 The request sent by the program file 28 includes a presentation space and a function command. The function command may include a keyboard input such as the <Enter> key, the <Home> key, or other key. The presentation space is a buffer area for storing screen characters or data. The size of the presentation space is defined to accommodate any fields which may be presented on a host screen. For example, if the host screens in a host session are 24 characters by 80 characters (24x80) then each  
10 screen may hold a maximum of 1920 characters. Accordingly, the presentation space will be defined to present a maximum of 1920 characters for each screen. An example of a presentation space is shown in FIG. 8. As shown in FIG. 8, the presentation space 80 displays a signon screen for establishing a host session on the host server 4 requesting a user id and a current password. Returning now to FIG. 7, the request sent  
15 by the program file 28 may include the presentation space 80 along with a user id, a password, and an <Enter> function command for logging on to the host server 4.

After sending the request to the host server 4, the logical operations 700 continue at operation 706 where the program file 28 receives a host response including an updated presentation space or host screen, in a data stream from the host server 4.  
20 Next, at operation 708, the program file 28 identifies the type of host response which was received. The program file 28 identifies the host response by accessing screen data in the properties files 30, 32A, and 32B and comparing the screen data to the updated presentation space or host screen.

As discussed above with respect to FIG. 4, the screen properties files 32A and  
25 32B include a responses section which includes the possible responses types (i.e., success, analyze, and reject) which may be returned for a screen during a host session as well as identifying text for each type of response. In identifying a response, the program file 28 compares text in a screen received in the host response to identifying text in a predefined screen properties file for the received screen to determine the type  
30 of response.

Depending on response type identified for the host response screen at operation 708, the program file 28 may perform a number of actions. At operation 710, if the program file 28 identifies the response type as "success," the program file 28 will send the host screen to the application program 26 which will process the host screen presentation space data for display on the display screen 23 of the client computer 2 at operation 712. At operation 714, if the program file 28 identifies the response type as "analyze," the program file 28 will analyze the host screen to determine the appropriate response type at operation 716. At operation 718, if the program file 28 identifies the response type as reject, the program file 28 notifies the application program 26 that an error has occurred and prints the host screen presentation space to an errors file as well as the field definitions defined for the host screen in its screen properties file at operation 720. This information may be then accessed by a user to assist in a determination of the cause of the error, which may be corrected by modifying or appending a properties file.

Referring again to FIGS. 8 and 10, an example for identifying a host response will now be described according to an illustrative embodiment of the invention. If the presentation space returned in the host response is a host signon screen (such as the one shown in FIG. 8) the program file may identify the host screen by comparing the host screen name (or other identifying text on the screen) to the screen name or identifying text property for the predefined version of the host screen in the common properties file 30. Once the host screen is identified, the program file 28 may identify the type of response by searching for identifying text in the response section defined for the identified host screen. Thus, if the host response screen contains the text "Your Password Must Be Supplied," (FIG. 11) the program file 28 will recognize this text as identifying a "reject" type of response. Once the response type is identified as reject, the program file 28 will print the host signon screen presentation space and the fields described in the associated screen properties file to an errors file.

It will be appreciated by those skilled in the art that the management of host sessions on a remote computer is facilitated by defining host screen data in properties files accessible by a client computer. Thus, when an error is detected during a host

session, the error may be corrected by modifying or appending the screen data in the properties files without recompiling code or re-recording changed screens.

5        Although the invention has been described in language specific to computer structural features, methodological acts and by computer readable media, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific structures, acts or media described. Therefore, the specific structural features, acts and mediums are disclosed as exemplary embodiments implementing the claimed invention.

10        The various embodiments described above are provided by way of illustration only and should not be construed to limit the invention. Those skilled in the art will readily recognize various modifications and changes that may be made to the present invention without following the example embodiments and applications illustrated and described herein, and without departing from the true spirit and scope of the present invention, which is set forth in the following claims.